

Ambiente Cliente-Servidor

Projeto Prático

1. Introdução

Pretende-se que os alunos tomem conhecimento com a programação em sockets e desenvolvam uma arquitectura de sistemas cliente-servidor.
O trabalho pode ser realizados em qualquer sistema operativo, utilizando ambientes de desenvolvimento da Linguagem Java. Para apoio à realização do exercício proposto deve utilizar as referências apresentadas no final deste documento. Este trabalho deverá ser realizado individualmente ou em grupos de dois ou três alunos.

2. Realização do Trabalho

Este trabalho tem como objectivo o desenvolvimento de uma solução de rede Cliente-Servidor (Figura 2). A solução a implementar deverá suportar múltiplos utilizadores em ambiente “multi-thread”. Os utilizadores não efetuam login, sendo identificados pelo endereço IP utilizado na rede. A solução vai permitir aos utilizadores enviarem mensagens a outros utilizadores por intermédio do Servidor

3. Funcionalidades / Protocolo de Comunicação

Tenha em atenção que deverá definir com rigor o formato das mensagens da sua aplicação, que deverá oferecer as seguintes mensagens:

3.1 Cliente-Servidor

No sentido cliente-servidor, deverão existir pelo menos as seguintes mensagens:

- obter lista branca de utilizadores;
- obter lista negra de utilizadores;
- listar utilizadores online;
- envio de mensagem para todos os utilizadores online;
- envio de mensagem para um determinado utilizador online;
- desconectar.

3.2 Servidor-Cliente

No sentido servidor-cliente, deverão existir pelo menos as seguintes mensagens:

- IP válido ou inválido
- listar lista branca de utilizadores;
- listar lista negra de utilizadores;
- enviar mensagem para um utilizador online;
- enviar mensagem para todos os utilizadores online;

As mensagens de resposta do servidor devem indicar sucesso ou insucesso da operação, por exemplo OK ou ERRO.

4. Desenvolvimentos

4.1 Servidor

Implementar um servidor que utilize sockets TCP/UDP e um protocolo de comunicações específico. O formato de mensagens da aplicação é definido pelo(s) aluno(s) de acordo com o ponto 3. Inicie o servidor por defeito no porto 6500 para comunicações TCP e envie mensagens para o porto 9031 no caso das comunicações UDP. O envio de mensagem para todos os utilizadores online ou para um determinado utilizador online deverá utilizar unicast UDP. O servidor terá que implementar "multi-threading", servindo ao mesmo tempo mais de um cliente.

O servidor Back-End deverá manter em ficheiro(s) a seguinte informação:

- Uma "lista branca" de endereços IPV4 ou classes de endereços válidos para os utilizadores válidos (ex: 192.168.10.10, 192.168.20.0/24).
- Uma "lista negra" de endereços IPV4 com utilizadores a rejeitar (ex: 192.168.10.21, 192.168.20.22).

Caso a lista branca esteja vazia, não haverá validação de endereços IP dos utilizadores, no entanto serão rejeitados todos os clientes da lista negra. Caso o mesmo IP esteja em ambas as listas (branca e negra), a lista negra terá prioridade.

O servidor deve terminar e imprimir na consola mensagens de erro caso sejam detectados erros nos ficheiros de dados. Para todas actividades de comunicação com os clientes deve haver como opção imprimir na consola do servidor e/ou guardar num ficheiro de log.

4.2 Cliente TCP/UDP

Implementar um cliente multi-thread que utilize sockets TCP/UDP e um protocolo de comunicações específico utilizado pelo Servidor, de acordo com o protocolo estabelecido no ponto 3. Utilize por defeito o porto 6500 para comunicações TCP com o servidor, e o porto 9031 para comunicações unicast UDP.

O cliente estabelece uma comunicação TCP com o servidor e inicia um cliente UDP. Os utilizadores não enviam mensagens UDP directamente para outros clientes, fazem-no sempre por intermédio do servidor. Veja-se um exemplo de execução do código cliente na Figura 1.

```
MENU CLIENTE

0  - Menu Inicial
1  - Listar utilizadores online
2  - Enviar mensagem a um utilizador
3  - Enviar mensagem a todos os utilizadores
4  - lista branca de utilizadores
5  - lista negra de utilizadores
99 – Sair

Opcao? 1
Utilizadores online:
0 – 192.168.10.10
1 - 192.168.10.12
2 - 192.168.10.15
3 – 192.168.10.22
4 – 192.168.10.110

Opcao? 2
Utilizador? 3
mensagem? mensagem de teste para um utilizador
OK, mensagem enviada a 192.168.10.22

Opcao? 3
mensagem? mensagem de teste para todos
OK, mensagem enviada a todos os utilizadores

Opcao? 1
Utilizadores online:
0 – 192.168.10.10
1 - 192.168.10.12
2 – 192.168.10.15
3 – 192.168.10.22

Opcao? 5
Lista negra:
192.168.10.125

Opcao? 99
A Sair
Cliente Desconectado..
```

Figura 1 : Exemplo de execução do cliente

5. Tratamento de excepções

A solução deverá ser tolerante a falhas temporárias de curta duração no canal de comunicação TCP. Ou seja, se a rede ou o servidor ficarem indisponíveis por alguns instantes, a sua aplicação irá receber uma exceção. Esta exceção deverá ser tratada do lado do cliente de forma a tentar de novo a abertura do socket com o servidor. O cliente não pode visualizar a ocorrência de excepcções na sua aplicação. Quando não existe conexão com o servidor o cliente deverá simplesmente indicar que aguarda ligação ao servidor. Se ao fim de algum tempo (ex: 15 segundos) o cliente continuar sem conexão para o servidor, deverá considerar a ligação como perdida.

Para simular estas situações poderá terminar o servidor e voltar a iniciá-lo. Tenha em

atenção que durante o período de tempo em que a conexão com o servidor esteja inactiva todas as mensagens enviadas pelos clientes e os dados introduzidos pelos utilizadores devem ser mantidas em buffers locais e não devem ser perdidas, sendo enviadas para o servidor assim que este voltar a estar activo.

6. Interface Gráfica

A solução utiliza linha de comando para clientes e servidores. O desenvolvimento de interfaces gráficas elaboradas não será bonificado.

7. Arquitectura da Solução

A arquitectura Cliente-Servidor a implementar será composta por um Servidor que recebe comandos dos Clientes vía sockets TCP e sockets UDP para enviar mensagens aos Clientes. Uma lista branca/negra existente no Servidor define IPs válidos/inválidos. Apenas os Clientes com IPs válidos podem comunicar com o Servidor. Os Clientes implementam sockets TCP para enviar comandos ao Servidor e sockets UDP para receber mensagens UDP enviadas pelo Servidor. O software Cliente permite assim aos utilizadores enviarem mensagens a outros utilizadores por intermédio do Servidor, utilizando para o efeito um protocolo específico de comunicação.

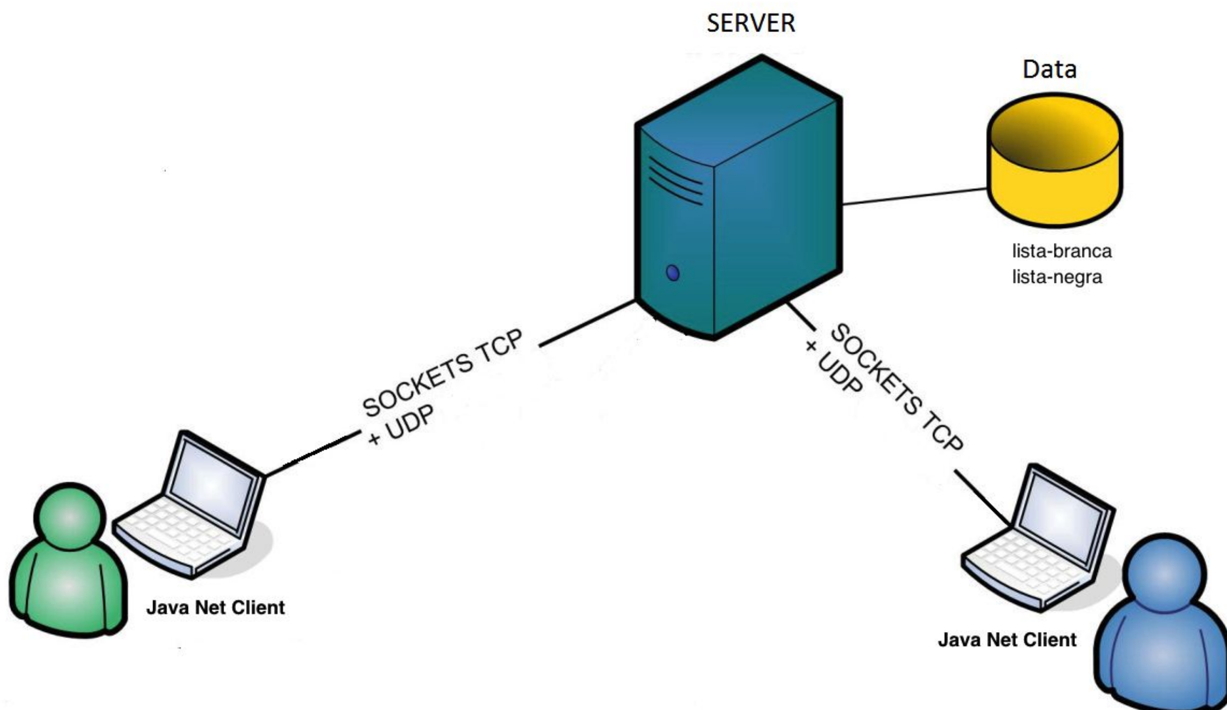


Figura 2: Arquitectura da solução

8. Relatório do Trabalho

Além do software desenvolvido deverá ainda escrever um relatório do trabalho prático. O referido relatório deverá conter os seguintes tópicos:

- Introdução;
- Arquitectura da aplicação;
- Descrição do protocolo de comunicação de nível aplicacional;
- Descrição dos testes efetuados à aplicação;

9. Entrega

A entrega será via Moodle no prazo de entrega aí definido, sendo a sua avaliação realizada em data/hora a divulgar. Não serão válidos trabalhos fora do prazo.
A entrega deverá utilizar um ficheiro zip, contendo o relatório (**obrigatório**) em PDF, bem como o código fonte dos programas realizados e ficheiros adicionais, obedecendo OBRIGATORIAMENTE ao seguinte formato:
RC-<nraluno1>-<nralunoN>-<>-projeto.zip

10. Referências

1. Java Networking, Tutorialspoint
   https://www.tutorialspoint.com/java/java_networking.htm

2. Networking - Java Tutorials, Oracle
   http://java.sun.com/docs/books/tutorial/networking/index.html